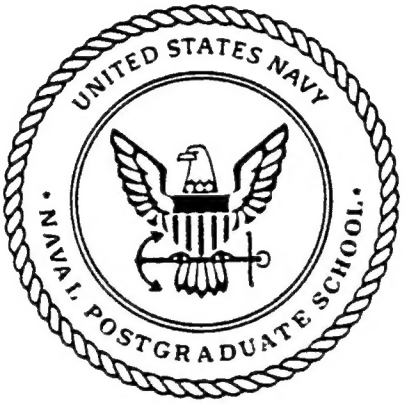
NPS-MA-95-001

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THE (IN)COMPLETE COMPENDIUM OF
COMPUTATIONAL CURIOSITIES:
GIVENS' ROTATIONS

by

C. F. Borges

Technical Report For Period
January 1995 - February 1995

Prepared for: Naval Postgraduate School
Monterey, CA 93943

NAVAL POSTGRADUATE SCHOOL
MONTEREY, CA 93943

Rear Admiral T.A. Mercer
Superintendent

Harrison Shull
Provost

This report was prepared in conjunction with research conducted for the Naval Postgraduate School and funded by the Naval Postgraduate School.

This report was prepared by:

Carlos F. Borges
Assistant Professor of Mathematics

Reviewed by:

RICHARD FRANKE
Chairman

Released by:

PAUL J. MARTO
Dean of Research

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION / AVAILABILITY OF REPORT
2b. DECLASSIFICATION / DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) NPS-MA-95-001	5. MONITORING ORGANIZATION REPORT NUMBER(S) NPS-MA-95-001
6a. NAME OF PERFORMING ORGANIZATION Naval Postgraduate School 6b. OFFICE SYMBOL (If applicable) MA	7a. NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943	7b. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943
8a. NAME OF FUNDING / SPONSORING ORGANIZATION Naval Postgraduate School 8b. OFFICE SYMBOL (If applicable) MA	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code) Monterey, CA 93943	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO / TASK NO / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)
The (In)Complete Compendium of Computational Curiosities: Givens' Rotations

12. PERSONAL AUTHOR(S)
Carlos F. Borges

13a. TYPE OF REPORT	13b TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Technical Report	FROM Jan 95 TO Feb 95	95 February 15	10

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Givens' Rotations

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

We look at the properties of Givens' rotations computed and performed in floating point arithmetic that conforms to the IEEE 754 standard. We describe an algorithm for constructing computationally orthogonal Givens' rotations.

20 DISTRIBUTION / AVAILABILITY OF ABSTRACT ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Carlos F. Borges	22b TELEPHONE (Include Area Code) 408-656-2124 22c. OFFICE SYMBOL MA/Bc

DD Form 1473, JUN 86 *Previous editions are obsolete.*
S/N 0102-LF-014-6603

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

The (In)Complete Compendium of Computational Curiosities: Givens' Rotations

CARLOS F. BORGES
Code Ma/Bc
Naval Postgraduate School
Monterey, CA 93943

February 15, 1995

Abstract

We look at the properties of Givens' rotations computed and performed in floating point arithmetic that conforms to the IEEE 754 standard. We describe an algorithm for constructing *computationally* orthogonal Givens' rotations.

1 Basic Concepts

We begin with the following definitions:

- $\mathcal{F}_p$ The set of normalized p-digit binary floating point numbers where the exponent range is assumed to be infinite (i.e. no overflow or underflow can occur).
- $fl(x)$ For $x \in \Re$, this is the closest element of $\mathcal{F}_p$ to x, with the last binary digit 0 in case of a tie.

We assume throughout that the operations $+, -, *, /$,and $\sqrt{}$ are correctly rounded.

2 The Classical Given's Roatation

Let $\mathbf{x} = [x_1 x_2]^T$ be an element of $\Re^2$. The classical Givens' rotation is an orthogonal matrix

$$G = \begin{bmatrix} c & s \\ -s & c \end{bmatrix}$$

such that

$$G\mathbf{x} = \begin{bmatrix} \|\mathbf{x}\| \\ 0 \end{bmatrix} \tag{1}$$

Givens' rotations are usually thought of geometrically as *rotation* matrices. Indeed, many derivations are given in terms of the geometry of the vector $\mathbf{x}$ in $\Re^2$ but the author finds that these lead to more confusion than understanding. Another way of writing a Givens' rotation is

$$G = \frac{1}{\|\mathbf{x}\|} \begin{bmatrix} x_1 & x_2 \\ -x_2 & x_1 \end{bmatrix}$$

Clearly the structure of G guarantees that equation 1 will be satisfied (at least in exact arithmetic).

Givens' rotations are useful because they are orthogonal, $G^T G = I$. In particular

$$G^T G = \begin{bmatrix} c^2 + s^2 & cs - sc \\ sc - cs & c^2 + s^2 \end{bmatrix}$$

If the *Pythagorean theorem* ($c^2 + s^2 = 1$) applies to the elements, then we have an orthogonal matrix (Note that the structure of the matrix guarantees that the off-diagonal elements are zero, even in floating-point arithmetic since it is commutative). Indeed, the key to computing a Givens' transformation is the ability to normalize a vector in $\Re^2$.

The classical algorithm for constructing Givens' rotations ([1], p. 45) is

if $x_2 = 0$
  $c = 1$;
  $s = 0$;
elseif $|x_2| \geq |x_1|$
  $t = x_1/x_2$;
  $s = 1/\mathbf{sqrt}(1 + t^2)$;
  $c = s * t$;
else
  $t = x_2/x_1$;
  $c = 1/\mathbf{sqrt}(1 + t^2)$;
  $s = c * t$;
end

We propose that a better organization of this algorithm which costs no more is

if $x_2 = 0$
  $c = 1$;
  $s = 0$;
elseif $|x_2| \geq |x_1|$
  $t = x_1/x_2$;
  $s = \mathbf{sqrt}(1/(1 + t^2))$;
  $c = s * t$;

else
  $t = x_2/x_1$;
  $c = \mathrm{sqrt}(1/(1+t^2))$;
  $s = c * t$;
end

Experiments using extended precision arithmetic show that the second algorithm is, on average, slightly closer to the correctly rounded Givens' rotation than the first one.

Usually, the acme of a floating-point computation is to give a correctly rounded result. Before examining what this means here, consider the following example.

EXAMPLE: The Givens' rotation G such that

$$G \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \begin{bmatrix} \sqrt{2} \\ 0 \end{bmatrix}$$

is

$$G = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}$$

In double precision using the correctly rounded value of $1/\sqrt{2}$ one finds that the diagonal elements of $G^T G$ are roughly $1 + 2.220410^{-16}$. This is only one *ulp* from the correct value but the end result is that the computed Givens' rotation is not orthogonal, it does not preserve norms but rather increases them.

Note that, in this example, our proposed improvement yields a G where each of the elements is the correctly rounded element from the Givens' rotation while the classical one does not.

The underlying cause of the problem we see in the above example is not due to rounding error in computing the rotation, rather it is the inability of the floating point system to represent the rotation sufficiently well. In double precision IEEE 754 arithmetic there is no floating point number twice whose square is 1.

3 Computationally Orthogonal Givens' Rotations

We now propose an algorithm for computing a Givens' rotation which is *computationally* orthogonal. That is, the algorithm computes a matrix G such that $G^T G = I$ to full working precision, the computed matrix G will be close to the classically computed Givens' rotation in the Frobenius norm.

if $x_2 = 0$
  $c = 1$;
  $s = 0$;

elseif $|x_2| \geq |x_1|$
  $t = x_1/x_2$;
  $s = \mathbf{sqrt}(1/(1+t^2))$;
  $b = s^2$;
  $c = \mathbf{sqrt}(1-b)$;
else
  $t = x_2/x_1$;
  $c = \mathbf{sqrt}(1/(1+t^2))$;
  $b = c^2$;
  $s = \mathbf{sqrt}(1-b)$;
end

We will need to explore certain properties of correctly rounding floating point arithmetic to see why this works.

The following theorem about correctly rounded square roots is based on a conjecture of Gragg and will be useful in what follows. It is stated and proved for $1/4 \leq x < 1$ since we can use exact argument reduction on the exponent to put any floating point number in this range.

Theorem 1 *Let $x \in \mathcal{F}_p$ be a floating point number such that $1/4 \leq x < 1$, and let $y = fl(\sqrt{x})$ be the correctly rounded floating point representation of $\sqrt{x}$. Then $fl(y^2)$ is within 1 ulp of x.*

Proof. It is clear that $1/2 \leq \sqrt{x} < 1$, and hence, since y is correctly rounded it follows that $y = \sqrt{x} + \delta$ where $|\delta| \leq 2^{-(p+1)}$. The exact square of y is

$$y^2 = x + 2\delta\sqrt{x} + \delta^2.$$

Clearly, if y^2 and x differ by less than $3/2$ of an ulp of x then so will $fl(y^2)$ and x because of correct rounding. So we need only show that $|2\delta\sqrt{x} + \delta^2|$ satisfies this bound. Of course

$$|2\delta\sqrt{x} + \delta^2| \leq |2\sqrt{x} + \delta| 2^{-(p+1)}.$$

We consider two cases. First, if $1/2 \leq x < 1$ then $\sqrt{x} \leq \sqrt{1 - 2^{-p}} < 1 - 2^{-(p+1)}$ and it follows that

$$\begin{aligned} |2\sqrt{x} + \delta| &\leq 2\sqrt{x} + |\delta| \\ &< 2 - 2^{-p} + 2^{-(p+1)} < 2. \end{aligned}$$

And hence

$$|2\delta\sqrt{x} + \delta^2| < 2^{-p}$$

which is less than 1 ulp of x.

Second, if $1/4 \leq x < 1/2$ then $\sqrt{x} \leq \sqrt{1/2 - 2^{-(p+1)}} < 1/\sqrt{2} - 2^{-(p+2)}$ and it follows that

$$|2\sqrt{x} + \delta| \leq \sqrt{2} < 3/2$$

And hence

$$|2\delta\sqrt{x} + \delta^2| < \frac{3}{2}2^{-(p+1)}$$

which is less than 3/2 ulps from x.
□

Theorem 2 *Let $x \in \mathcal{F}_p$ be a floating point number such that $1/2 \leq x \leq 1$, then the difference $1 - x = fl(1 - x)$. Moreover, the last bit of the binary representation of $fl(1 - x)$ is 0.*

Proof. If $x = 1$ or $x = 1/2$ the result is obvious, so we look at the case where $1 > x > 1/2$. Let $d = x - 1/2$. Then $1 - x = 1 - (1/2 + d) = 1/2 - d$. The binary expansion for d must look like $.0bbb...bb$ where there are p bits in the expansion, the first is zero, and the rest can be either zeros or ones, but at least one of them is a one. Clearly now we are computing

$$\begin{array}{rr} & .1000...00 \\ - & .0bbb...bb \end{array}$$

which is a p-bit number whose first bit must be zero since the result is less than 1/2. Hence, upon normalization the exactly computed result is an element of $\mathcal{F}_p$ with its last bit a zero. □

4 Why does it work?

Let us consider the case where the following code fragment is executed

elseif $|x_2| \geq |x_1|$
$t = x_1/x_2;$
$s = \mathbf{sqrt}(1/(1 + t^2));$
$b = s^2;$
$c = \mathbf{sqrt}(1 - b);$

In this case $0 \leq t \leq 1$ and thus $1/2 \leq 1/(1 + t^2) \leq 1$. After taking the square root we square s and make an assignment to force a rounding, b should be a number between 1/2 and 1. It is possible for b to be slightly less than 1/2 at this point, we will discuss this later. For now assume $b > 1/2$. We now compute $c = \sqrt{1 - b}$, there is no reason to force an intermediate rounding for $1 - b$ with an assignment since this is computed exactly anyway. Now note that from theorem 1 we know that $c^2 = (1 - b) \pm 1ulp$ where the error is one ulp of b. Let $.sss...ss$ be binary expansion of $fl(s^2)$, and let $.0bb...bb0$ be the binary expansion of $1 - b$, then, in the worst case, the binary expansion of $fl(c^2)$ must be $.0bb...bb0 \pm .000...00\delta$, where δ can be either 0 or 1, it will be more usual that the δ will be further to the right. Then, in binary, $c^2 + s^2$ must be

$$
\begin{array}{rl}
 & .sss...ss \\
+ & .0bb...bb0 \\
\pm & .000...00\delta
\end{array}
$$

The sum is either 1, or it is 1.000...001 which rounds to 1 by the round to even rule, or .111...111 which also rounds to 1 by the round to even rule. If the δ is further to the right then simple rounding will give us a 1.

There is a problem if b is less than $1/2$. This happens in IEEE 754 single precision calculations since the square of the square root of $1/2$ here is less than $1/2$. This only occurs if $1/(1+t^2)$ is sufficiently close to $1/2$ so one can avoid it with a simple trap. This is not a problem in IEEE 754 double precision calculations and the code will work as written.

One can make a variation on this algorithm which preserves the computational orthogonality but reduces the Frobenius norm of the difference between the computationally orthogonal Givens' rotation and the classically computed one. To do this first compute the classical Givens' rotation. Then check to see if it is computationally orthogonal. If it is then stop. Otherwise compute the computationally orthogonal rotation. Then use bisection on the smaller of c or s to find the computationally orthogonal rotation which is closest to the classical one. It is not clear that one would want to do this. The algorithm we have given tries to find a computationally orthogonal rotation that is as close to a truly orthogonal rotation as possible. Using a bisection as just described would take you away from this point.

5 Another Algorithm for Rotations

The following is yet another variation on the theme of orthogonal rotations. It uses techniques from simulated extended precision (SEP) arithmetic to do its work and is written for IEEE 754 double precision arithmetic (although it can be modified to work in single precision). Experiments with random vectors from $\Re^2$ with uniformly distributed elements show that it is computationally orthogonal roughly 95% of the time. These experiments also show that when the rotation is not exactly orthogonal then the diagonal of G^TG contains the number directly to the left of 1 in the floating point system.

if $x_2 = 0$
  $c = 1;$
  $s = 0;$
elseif $|x_2| \geq |x_1|$
  $t = x_1/x_2;$
  $s = \mathbf{sqrt}(1/(1+t^2));$
  $bb = 134217728 * s;$
  $b = (s - bb) + bb;$

$bb = s - b;$
$tmp = b^2;$
$dd = 2 * b * bb;$
$d = tmp + dd;$
$dd = ((tmp - d) + dd) + bb^2;$
$b = 1 - d;$
$b = b - dd;$
$c = sign(t) * \mathbf{sqrt}(b);$
else
$t = x_2/x_1;$
$c = \mathbf{sqrt}(1/(1 + t^2));$
$bb = 134217728 * c;$
$b = (c - bb) + bb;$
$bb = c - b;$
$tmp = b^2;$
$dd = 2 * b * bb;$
$d = tmp + dd;$
$dd = ((tmp - d) + dd) + bb^2;$
$b = 1 - d;$
$b = b - dd;$
$s = sign(t) * \mathbf{sqrt}(b);$
end

References

[1] G.H. Golub and C.F. Van Loan. *Matrix Computations*, Johns Hopkins University Press, 1983.

DISTRIBUTION LIST

Director (2)
Defense Tech Information Center
Cameron Station
Alexandria, VA 22314

Research Office (1)
Code 81
Naval Postgraduate School
Monterey, CA 93943

Library (2)
Code 52
Naval Postgraduate School
Monterey, CA 93943

Professor Richard Franke (1)
Department of Mathematics
Naval Postgraduate School
Monterey, CA 93943

Dr. Richard Lau (1)
Office of Naval Research
800 Quincy St.
Arlington, VA 22217

Dr. William Gragg, Code MA/Gr (1)
Department of Mathematics
Naval Postgraduate School
Monterey, CA 93943